PROJECT REPORT ON PLASMA DONOR APPLICATION

Date	07 November 2022
Team ID	PNT2022TMID31589
Project Name	Plasma Donor Application

TEAM MEMBERS:

Team Lead : Jila Rasnat B

Team Member 1 : Blessy Jefrina H

Team Member 2 : Karthikeyan B

Team Member 3 : Lakshmi Meena M

Team Member 4 : Eisha V S

CONTENTS

1. **INTRODUCTION**

   1.1 Project Overview
   1.2 Purpose

2. **LITERATURE SURVEY**

   2.1 Existing Problem
   2.2 References
   2.3 Problem Statement Definition

3. **IDEATION AND PROPOSED SOLUTION**

   3.1 Empathy Map Canvas
   3.2 Ideation And Brainstorming
   3.3 Proposed Solution
   3.4 Problem Solution fit

4. **REQUIREMENT ANALYSIS**

   4.1 Functional requirement
   4.2 Non-Functional requirements

5. **PROJECT DESIGN**

   5.1 Data Flow Diagrams
   5.2 Solution & Technical Architecture
   5.3 User Stories

6. **PROJECT PLANNING AND SCHEDULING**

   6.1 Sprint Planning And Estimation
   6.2 Sprint Delivery Schedule
   6.3 Reports from JIRA

7. CODING AND SOLUTIONING

7.1 Feature 1
7.2 Feature 2
7.3 Feature 3
7.4 Database Schema (if Applicable)

8. TESTING

8.1 Test Cases
8.2 User Acceptance Testing

9. RESULTS

9.1 Performance Metrics

10. ADVANTAGES AND DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

13.1 Source Code
13.2 GitHub And Project Demo Link

1. INTRODUCTION

1.1 Project Overview

A plasma is a liquid portion of the blood, over 55% of human blood is plasma. Plasma is used to treat various infectious diseases and it is one of the oldest methods known as plasma therapy. Plasma therapy is a process where blood is donated by recovered patients in order to establish antibodies that fights the infection. In this project plasma donor application is being developed by using Python, Flask and Dockers. The services used are Kubernetes, Flask, Docker and the mails are sent to the users using Send Grid. For instance, during COVID 19 crisis the requirement for plasma increased drastically as there were no vaccination found in order to treat the infected patients, with plasma therapy the recovery rates where high but the donor count was very low and in such situation it was very important to get the information about the plasma donors. Saving the donor information and notifying about the current donors would be a helping hand as it can save time and help the users to track down the necessary information about the donors. In the recommendation system, the donor who wants to donate plasma can donate by uploading their COVID19 certificate and the blood bank can see the donors who have uploaded the certificate and they can make a request to the donor and the hospital can register/login and search for the necessary things. Plasma from a blood bank and they can request a blood bank and obtain plasma from the blood bank.

1.2 Purpose

The most reason of the proposed framework, the donor who wants to donate plasma can simply upload their covid19 traced certificate and can donate the plasma to the blood bank, the blood bank can apply for the donor and once the donor has accepted the request, the blood bank can add the units they need and the hospital can also send the request to the blood bank that urgently needs the plasma for the patient and can take the plasma from the blood bank.

The main purpose/objective of our project are:

- To build a platform between plasma donor and receiver.
- To find the nearest plasma donor in a specific region in the shortest possible time.
- To increase the number of voluntary unpaid plasma donations significantly.

2. LITERATURE SURVEY

2.1 Existing Problem

The people who donate plasma to patients are not checked properly. If the donor has any possible medical problem and donate plasma to the recipient, the danger may arise. One who donates plasma should always be verified of the donor's medical records.

- Cannot Upload and Download the latest updates.
- No use of Web Services and Remoting.
- Risk of mismanagement and of data when the project is under development.
- Less Security.
- No proper coordination between different Applications and Users.
- Fewer Users – Friendly

In the existing model, there's no algorithm to find the nearest donor during emergency cases. The user needs to register for the type of plasma needed and has to wait for the donor to arrive in case of unavailability of that type of plasma in the bank.

Medical histories would be like:

- No blood should be donated by an individual with anemia. People who suffer with anemia should not donate their Plasma (blood).
- Donors who have blood-borne illnesses should not apply for plasma donation.
- Only the people who has fully recovered from covid-19 infection can donate their plasma to help those are currently affected with covid-19.
- After a few years of pregnancy and during pregnancy, women do not donate plasma.

In the existing model, the above medical records are not employed. This could lead to impairment. Plasma does not necessarily match the donor and patient's body state the two things below are not taken into account the location of the donor as well as the distance between both the recipient and donor.

2.2 References

S.No	Title	Publication details	Methodology/ algorithms	Merits	Demerits
1.	Developing a plasma donor application using function-as-a-service in AWS.	Aishwarya R Gowri	Plasma therapy is a process where blood is donated by recovered patients in order to establish antibodies that fights the infection. In this project plasma donor application is being developed by using AWS services.	Saving the donor information and notifying about the current donors would be a helping hand as it can save time and help the users to track down the necessary information about the donors.	Cost expansive.
2.	Instant plasma donor recipient connector web application.	Kalpana Devi Guntoju, Tejaswini Jalli, Sreeja Uppala, Sanjay Mallisetti	In the absence of an approved antiviral treatment plan for a fatal COVID-19 infection, plasma therapy is an experimental approach to treat COVID-19 positive patients and help them for faster recovery.	They can request a blood bank and obtain plasma.	Lack of different bloodgroups.
3.	Nearest Blood & Plasma Donor Finding	NayanDas MD. Asif Iqbal	Recently a life-threatening virus, COVID-19, spread throughout the globe, which is more vulnerable for elder people and those with pre-existing	Closest blood or plasma donors of the same group in a particular area can be explored within less time and more efficiently.	However, a person who donates plasma may experience minor adverse effects, or any other procedure

			medical conditions. For them, plasma is needed to recover their illness.		involving a puncture, so risks are involved.
4.	Convalescent Plasma Therapy: Data driven approach for finding the Best Plasma Donors	Dr. G. Aghila, MNNoorshidha	The solution is based on (i)classification model - predict whether the donor has the threshold antibody level for donation (ii)regression model that can predict which donor can have a better level antibody titers in his plasma based on his/her clinical history.	This can prevent wastage of time, cost, especially, in emergencies.	During a plasma donation, a healthcare professional draws the blood from a vein. If they accidentally puncture an artery: 1.The blood will be bright red. 2.The blood will leave the body rapidly.
5.	Covid-19 Plasma Monitoring Based on Clustering a Large Set of Recovered Patient Data	Al-Rammahi Ali. A., Sari Farah, Al-Jelaihawi. Fahad. G	The increase in the number of cases of COVID-19 has led to huge data that needs to be clustered and classified so that it is more useful for tracking plasma for patients recovering from this disease.	Easy to track the cured patients.	The data need a complex algorithm.
6.	Determinants of plasma donation	Antoine Beurel, Florence Terrade	The need for plasma- derived products has been strongly increasing for some years, and blood collection	According to scientific study, regular donation of plasma and even whole blood has	1.Possible side effects of donating plasma include dehydration, vein damage,

			agencies have to adapt if they want to meet this demand.	health benefits to the donor.	fainting, and fatigue. 2.People who cannot recover properly often feel nauseous and sleepy, and may collapse for several hours.
7.	Implementation of blood donation application using android smartphone	Ms. Pradnya Jagtap, Ms. Monika Mandale, Ms. Prachi Mhaske, Ms.Sonali Vidhate, Mr.S.S. Patil	This project aims at maintaining all the information pertaining to blood donors, different blood groups available in each blood bank and helps them to manage in a better way.	Easy connecting donors and recipients makes blood donation way more proficient It connects blood donors and recipients through a single and scalable platform.	Less accuracy Connection between the user is complex.
8.	Blood Donor App Using Flutter for Blood Donation	R. Gokul, E. Nagul Vijay, R. Bharathi Raj, P. C. Thirumal	Blood donor app is an application that makes the process of searching for the blood in the emergency time and making the donor to reach the destination in a faster way. Making an analysis that the set of peoples who is already donating and who will be willing to make	The people who already donated can be reached, and who will be willing to make the next donation will be reached easily.	Tracking the donor is complex process in emergency conditions.

			the next donation.		
9.	Smart Blood Bank as a Service on Cloud	Bharathwaj Muralidaran, Akshay Raut, Yogesh Salve, Shivshankar Dange, Likhesh Kolhe	This project is basically focused on improving conventional working of blood bank management information system using the concept of cloud computing.	By Login and Register of the user the blood donor can be identified easily.	There is no security in the cloud database Can be lack of blood in emergency cases.
10.	Blood Bank App using Raspberry PI	Surabhi S. Pohandulkar Chhaya S. Khandelwa	The paper "Blood bank application using raspberry pi" is proposed to bring near blood bank and the person who need the blood due to accident or any emergency.	It connects blood donors and recipients through a single and scalable platform.	High complexity.

2.3 Problem Statement Definition

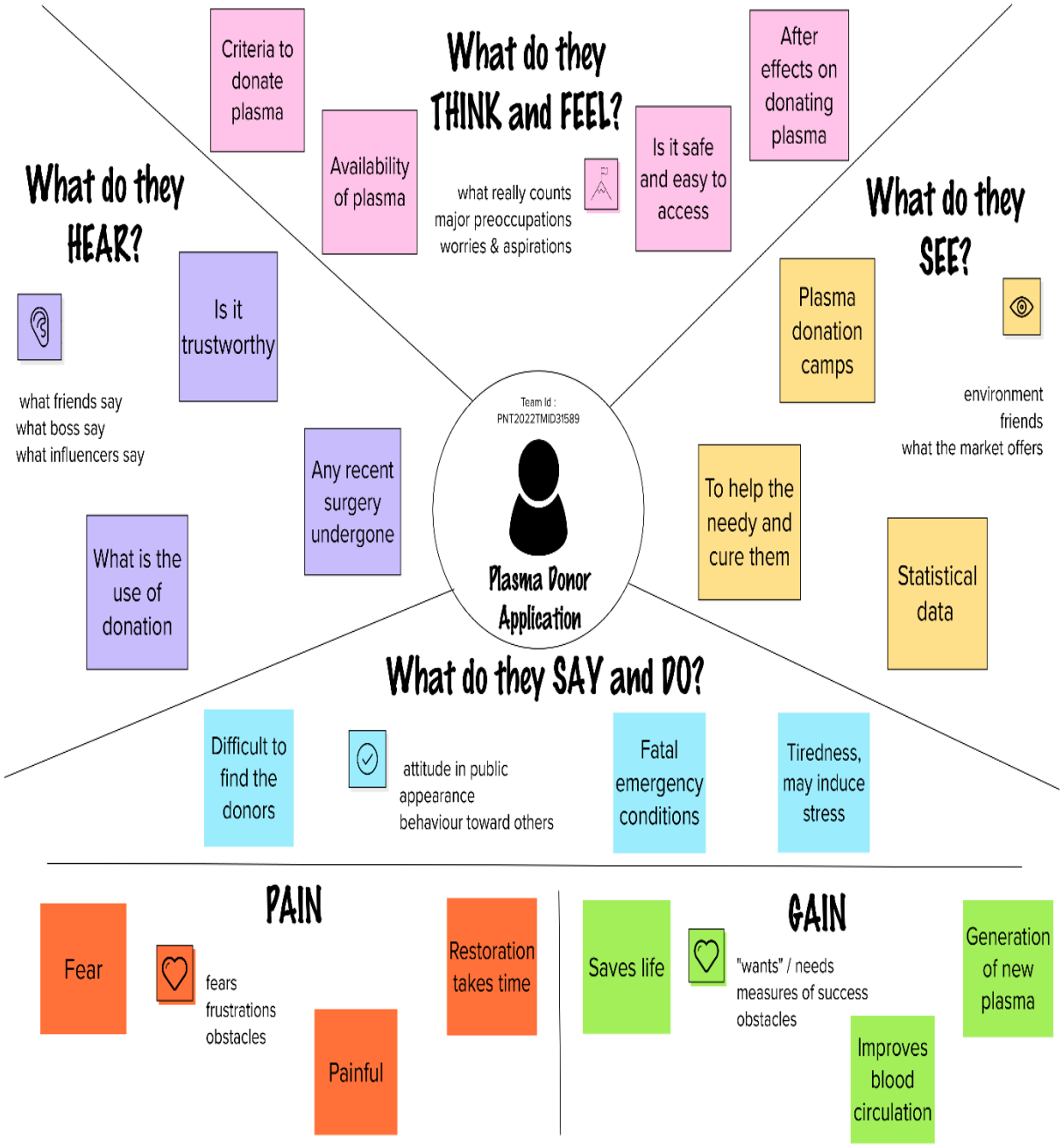
Plasma is the liquid component of blood and makes up 55% of the total blood volume. It is important for maintaining body functions. Plasma donation replaces lost blood and plasma in the body when a large amount of blood is lost due to surgery, accident, or immunodeficiency. During the COVID-19 crisis, the need for plasma became a higher priority and the number of donors decreased. A productive measure can be made to help the needy by saving donor information and keeping up-to-date donor lists. With this issue in mind, an application has been built to collect and store donor data and provide information on demand.

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	a donor	donate plasma	I do not know the procedure	I am unaware of plasma donation	upset
PS-2	a recipient	get plasma transfusion	I am unable to find the donors	I am unaware of the availability of plasma and its donor	distressed

3. IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canvas

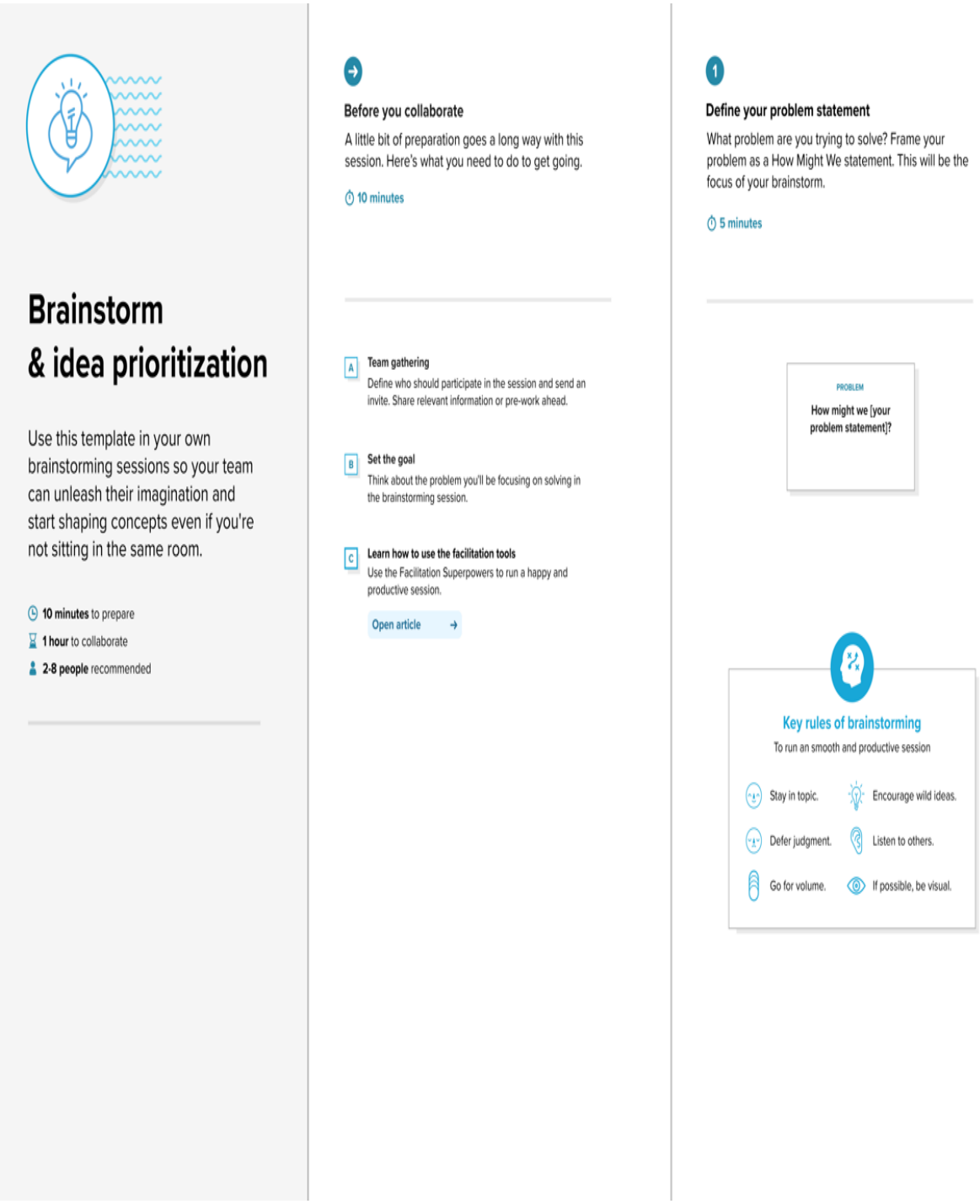
An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to helps teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.



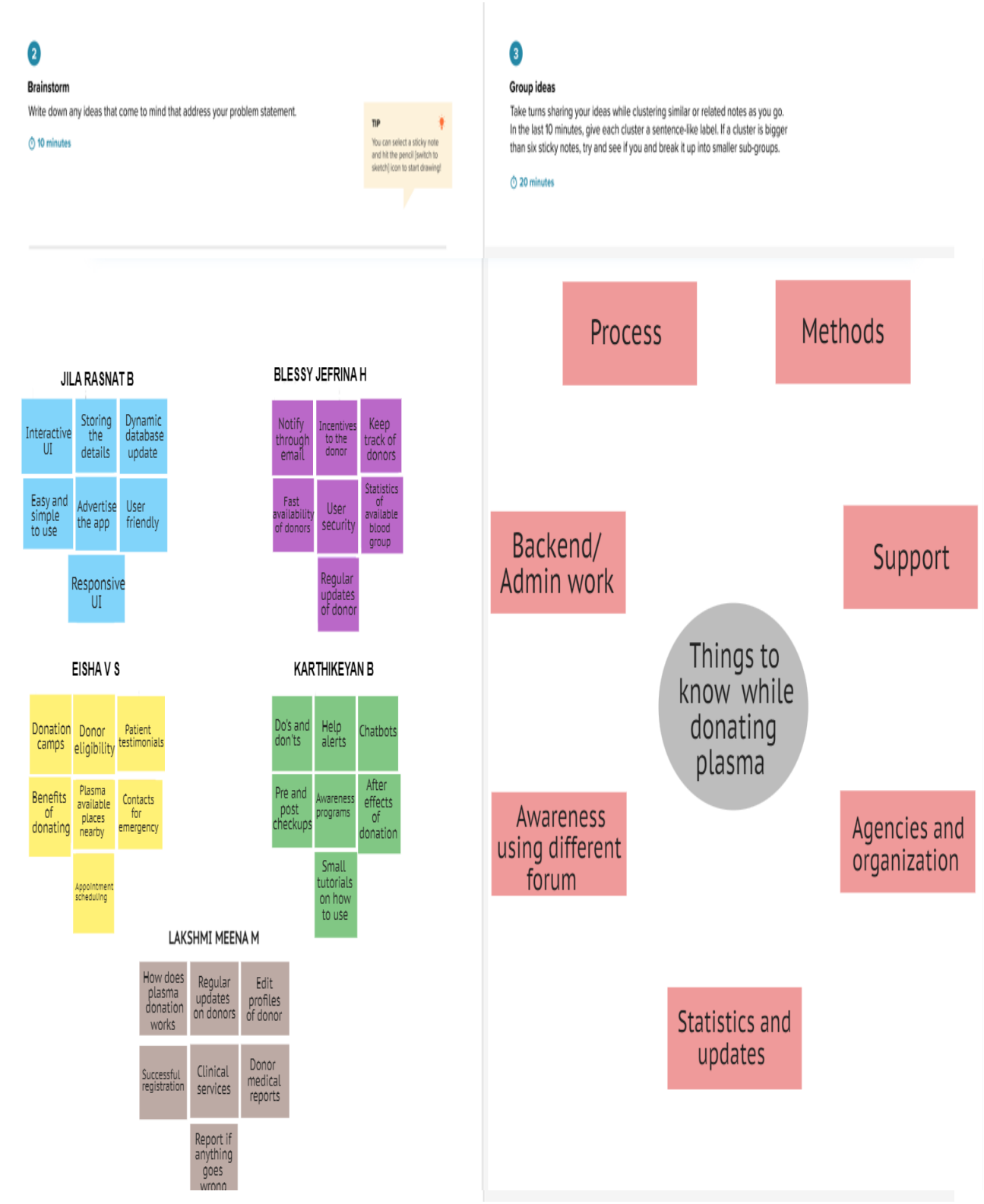
3.2 Ideation And Brainstorming

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Step-1: Team Gathering, Collaboration and Select the Problem Statement



Step-2: Brainstorm, Idea Listing and Grouping



Step-3: Idea Prioritization

4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

3.3 Proposed Solution

S. No	Parameter	Description
1.	Problem Statement (Problem to besolved)	During the COVID 19 crisis, the requirement of plasma became a high priority and the donor count has become low. Saving the donorinformation and helping the needy by notifyingthe current donors list, would be a helping hand.In regard to the problem faced, an application isto be built which would take the donor details, store them and inform them upon a request.
2.	Idea / Solution description	• The user interacts with theapplication. • Registers by giving the details as a donor and can request any blood groupin need or donation. • The database will have all the details and if a user posts a request, then the concerned blood group donors will getnotified about it. • The database contains all the details of the donation camp location and events.
3.	Novelty / Uniqueness	The simple visual user interface displays the availability of plasma or blood group data for donation. The user can check for the specific blood group needed and can request for plasma at any time according to their needs. The user will get an email notification indicating the availability of the plasma, also denoting whether it is available or in a short supply. The users can register and schedule anappointment in case of donation. The users who are volunteering for plasma donation will be provided with an e-certification.

4.	Social Impact / Customer Satisfaction	Many people would like to help those in need through donations of money, blood or plasma, but are unaware of technological innovations, including the use of social media, to identify and contact those in need which is difficult to take. Most of the facilities does not have access to patient's data to retrieve it quickly for any emergency conditions. A single platform to maintain all activities and information related to blood or plasma donation will increase the confidence of citizens to engage in these activities and to donate blood.
5.	Business Model (Revenue Model)	There is a free application for plasma donors. It is readily available and accessible to everyone. As it is difficult to find donors that match a particular blood type, this application allows users to register who they would like to donate plasma to and store that information in the database. By notifying current donors of the need to do so, it helps to preserve donor information. The need for plasma has increased significantly during the COVID-19 crisis and the number of donors is limited.
6.	Scalability of the Solution	The proposed solution helps the user to know the availability of plasma, to check compatibility of blood group, to register for donation and to request for the specific blood group in need. The application also sends an email notification if the requested blood groupis available. The user can also schedule an appointment if they are willing to donate. The donors are notified of the need for donations, making it easy to find donors at the right time.

3.4 Problem Solution fit

Project Title : Plasma Donor Application | Project Design Phase-I - Problem Solution Fit | Team ID: PNT2022TMID31589

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

(i)People willing to donate plasma.

(ii)Individuals in need of plasma.

6. CUSTOMER CONSTRAINTS CC

(i)Network connectivity.

(ii)Shortage of plasma.

(iii)Only registered users can donate and get information related to plasma.

5. AVAILABLE SOLUTIONS AS

(i)Plasma availability - Not up-to-date.

(ii)The customer can inform their queries through sending the mail.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

(i)The statistics should be updated often.

(ii)Customer couldn't know how to find the donor.

9. PROBLEM ROOT CAUSE RC

(i)More information about the plasma is available.

(ii)It is much easier to understand and navigate.

7. BEHAVIOUR BE

(i)A large amount of requests for plasma donation can be processed at the same time.

(ii)This application can collaborate with the Government and Non-Profitable Organizations.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

Ease of access and requirement of blood type.

4. EMOTIONS: BEFORE / AFTER EM

Before: Not sure to find nearest donors available.

After: Helps in finding the nearest donor.

10. YOUR SOLUTION SL

The location to where the plasma is needed will be given in mail. Donors can also schedule appointments to their convenience.

8. CHANNELS of BEHAVIOUR CH

The customer can register their request needs and when the donors are available the mail is sent to the customer.

When the plasma is not available the mail will be sent as not available.

Extract online & offline CH of BE

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

Following are the functional requirements of the proposed solution:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through form Registration through Gmail Registration through LinkedIn Registration through Facebook
FR-2	User Confirmation	Confirmation via OTP Email Confirmation
FR-3	User Dashboard	View and manage content in My Profile View donation history Download receipts
FR-4	Plasma request and donation	To check availability Details of donation/request made Distribution status
FR-5	User Search	Search available donors and location
FR-6	Admin Dashboard	View and manage donors information View and manage the plasma details

4.2 Non-Functional requirements

Following are the non-functional requirements of the proposed solution:

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The website needs to be easy to use even for anon-technical user.
NFR-2	**Security**	The entire system needs to be safe and protectedagainst malware and unauthorized attacks.
NFR-3	**Reliability**	The system needs to be reliable enough and needsto function without any technical failures.
NFR-4	**Performance**	The system needs to be as fast as possible regardless of the number of integrations and the traffic on thewebsite increases.

NFR-5	**Availability**	The system needs to be accessible to a user at anygiven point of time.
NFR-6	**Scalability**	The system needs to be scalable enough to support a large number of users at the same time whilemaintaining the optimal performance.

5. PROJECT DESIGN

5.1 Data Flow Diagram

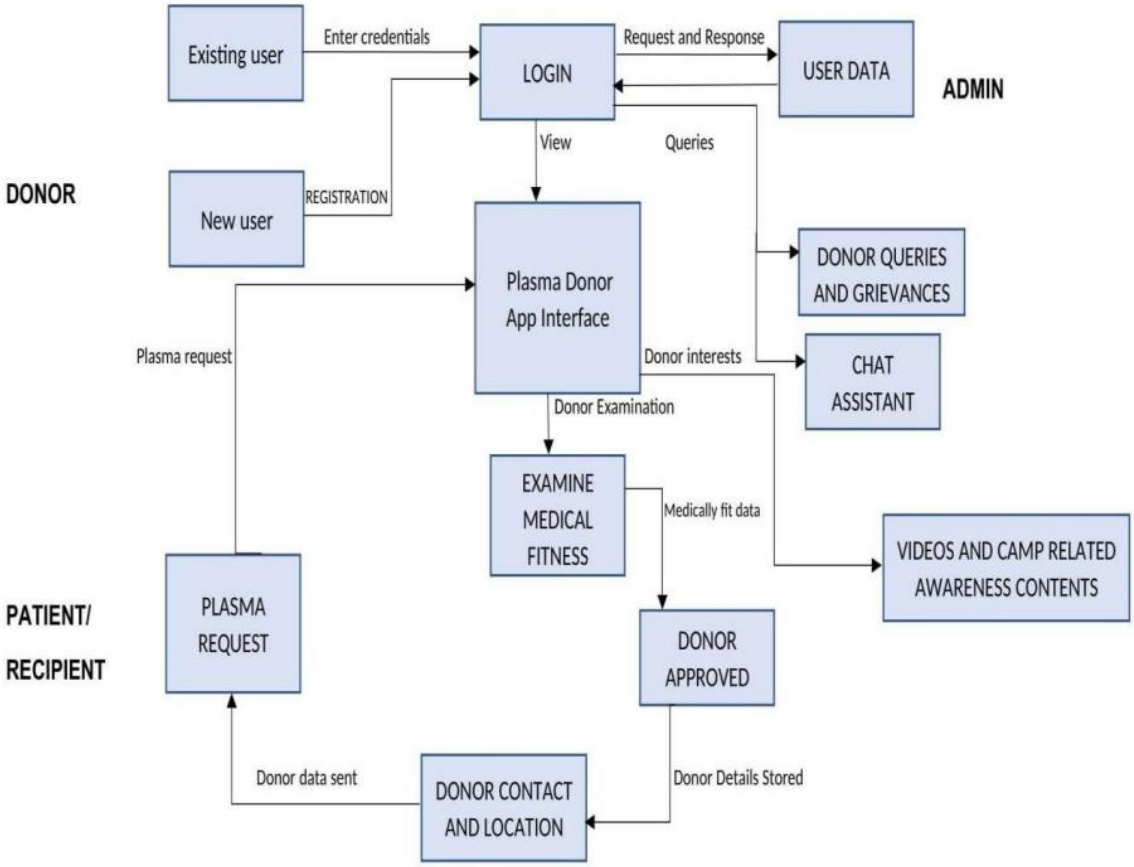


Fig: Data Flow Diagram

5.2 Solution & Technical Architecture

a) Solution Architecture Diagram

Solution architecture is a complex process – with many sub-processes – that bridgesthe gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of thesoftware to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed,and delivered.

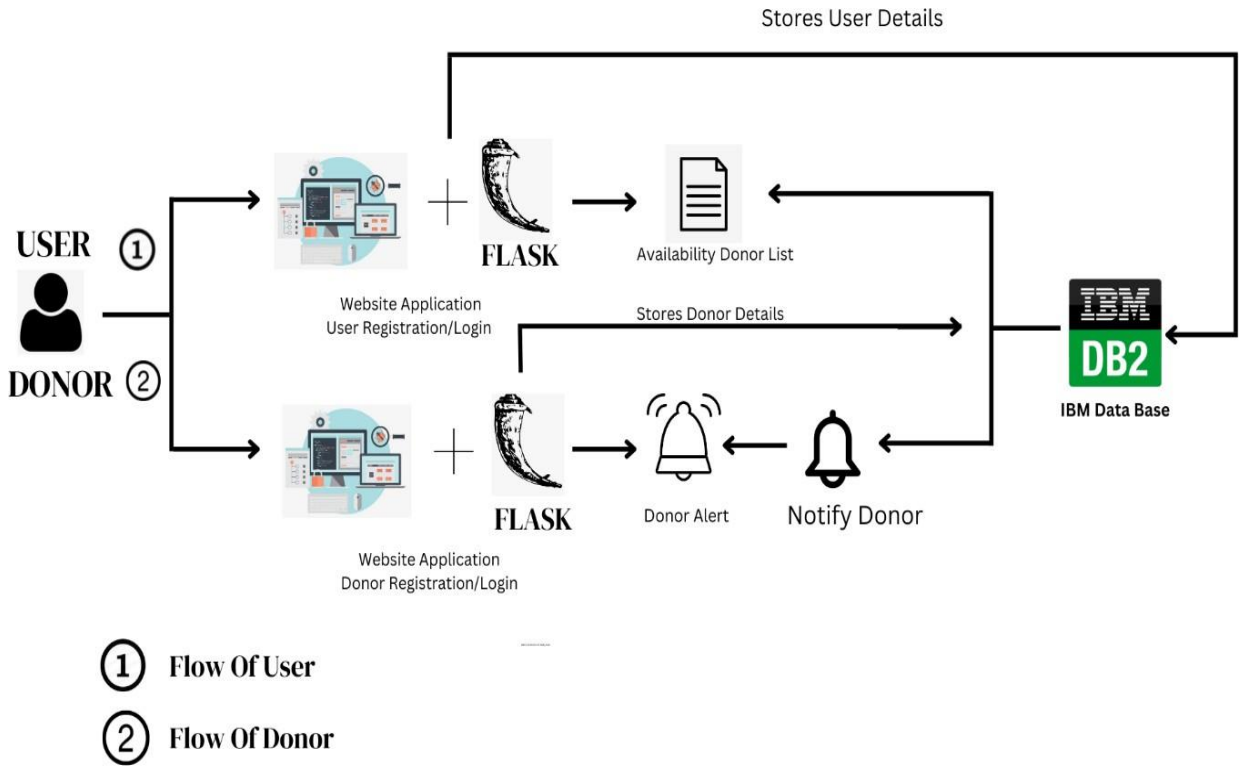


Fig: Architecture and data flow of the plasma donor application

b) Technical Architecture

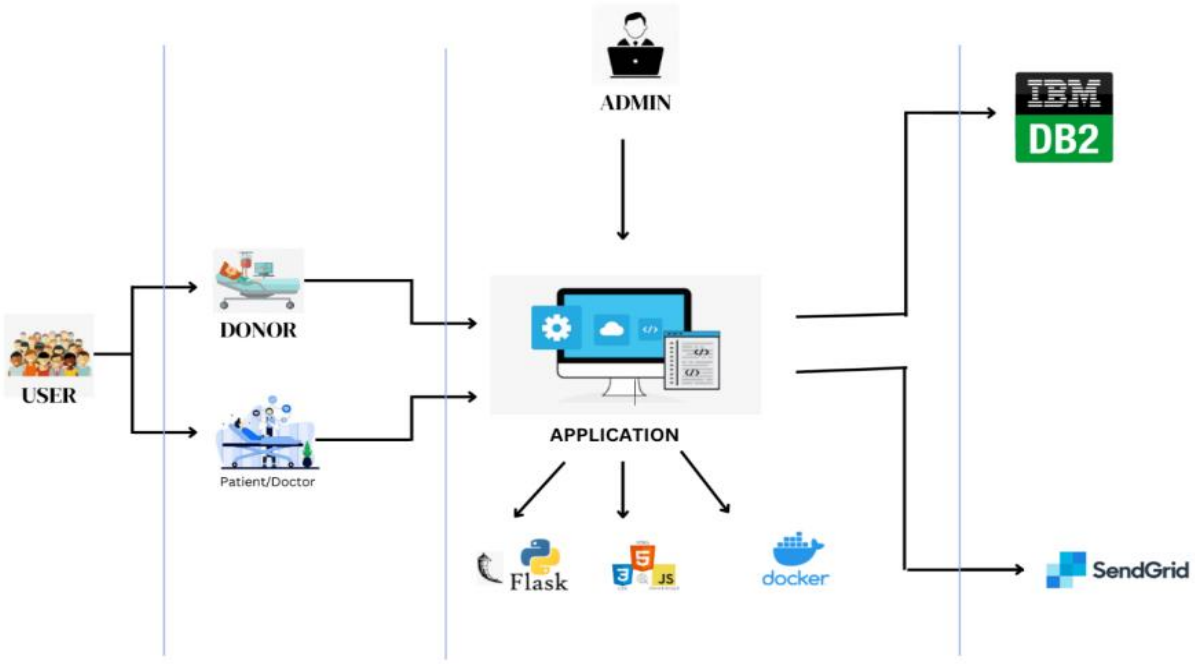


Fig: Technical Architecture

Table 1: Application Characteristics

S. No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Python – flask is an open-source framework used to develop the application.	Python - Flask
2.	Security Implementations	Container registry and Kubernetes Cluster areused for encryption of data.	Container registry and Kubernetes Cluster
3.	Scalable Architecture	Kubernetes Cluster allow containers to run across multiple machines and environments.	Kubernetes Cluster
4.	Availability	Kubernetes Cluster provides all timeavailability.	Kubernetes Cluster
5.	Performance	Docker improves the applicationperformance.	Docker

Table 2: Components and Technologies

S. No	Component	Description	Technology
1.	User Interface	The interaction between the user and application e.g., Web UI, Mobile App,Chatbot etc.	HTML, CSS, JavaScript / Bootstrapetc.
2.	Application Logic-1	Framework used for design the application.	Python, Python - Flask
3.	Application Logic-2	Accessing the cloud and storing the details of theusers both donors and patients.	IBM Cloud, IBM DB2
4.	Application Logic-3	Docker is an open-source platform for building,deploying, and managing containerized applications.	Docker
5.	Database	Data Type, Configurations etc.	SQL.
6.	Cloud Database	Database Service on Cloud	IBM Cloudant, IBM DB2 etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	External API-1	They make it easier for developers to store,manage and deploy container images.	Container Registry
9.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud	Local, Cloud Foundry, Kubernetes, etc.

5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user) Donor	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account/ dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Social media accounts	I can register & access the app with Social media account	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail other Email services	I can register the app with email account	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can register and access user profile with Gmail account	High	Sprint-1
Patient	Recipient	USN-6	As a requester, I can request the blood group for which I need plasma	I can get plasma from donors when available	High	Sprint-2
Customer (Web user) Donor	Profile	USN-7	As a user, I can see registration page, login page and chat bot for which the user can access to donate and to request for the required blood group plasma.	I can login through email and social media account for registration.	Medium	Sprint-2

Customer Care Executive	Help desk /User support for App	USN-8	As a helpdesk supporter, I can solve the queries and grievances of the user	I can reply to queries and give solutions to problems	High	Sprint-3
Administrator	Registration support	USN-9	As an admin, I can view the database of the registered user	I can check and verify the registered user's login credentials	Medium	Sprint-4
	Dashboard	USN-9	As an admin, I can manage plasma requests and other technical glitches in the app	I can check request numbers and troubleshoot problems in the app	Medium	Sprint-4
Chat Assistant	Dashboard	USN-10	In addition to customer care executive, I can help with user's queries within the app	I can reply to user's queries in the app	Medium	Sprint-4

6. PROJECT PLANNING AND SCHEDULING

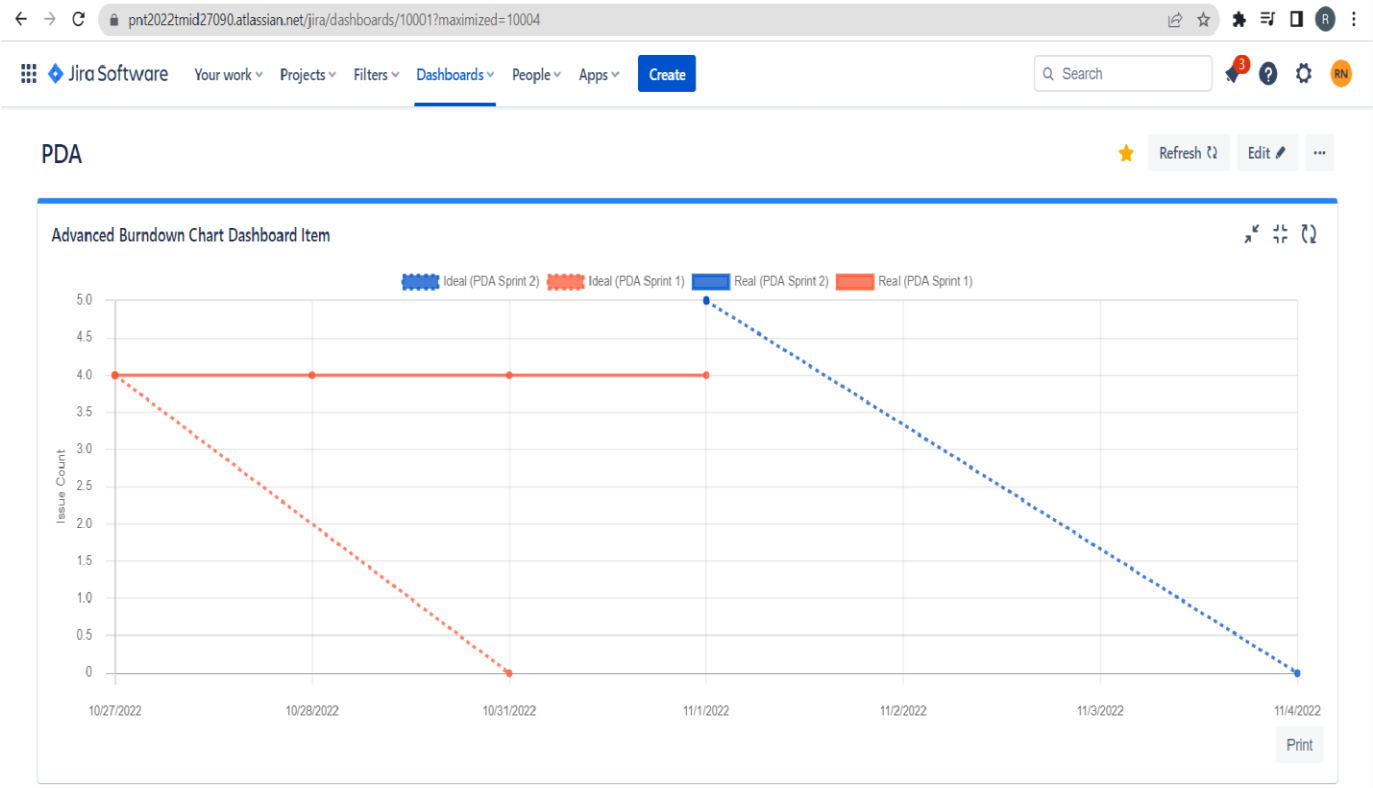
6.1 Sprint Planning And Estimation

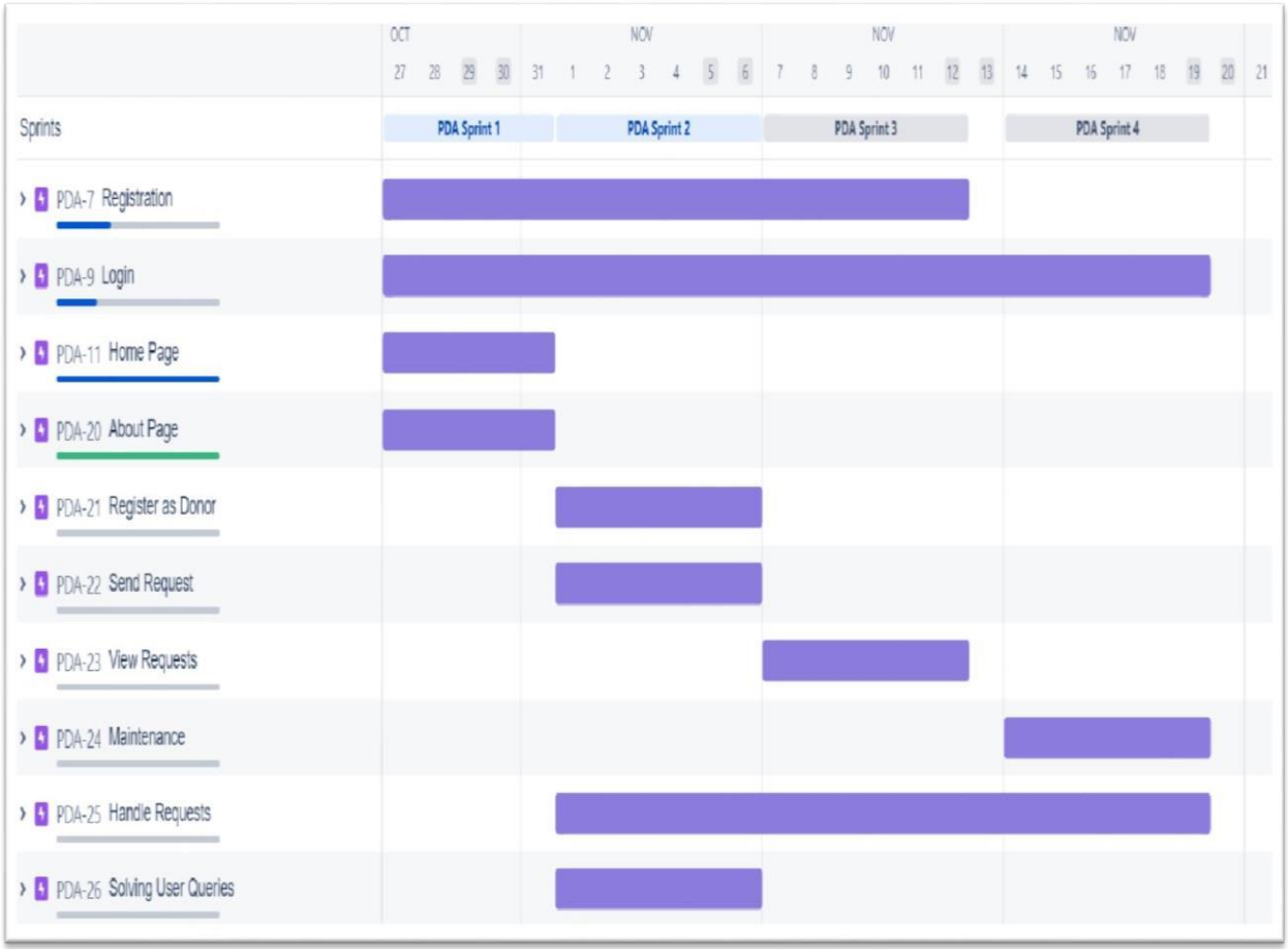
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Priority	Team Members
Sprint-1	Donor Registration	USN-1	As a user, I can register in the donor application by entering my name and other details.	High	Jila Rasnat B Lakshmi Meena M
Sprint-1	Login	USN-2	As an admin, I can log into the applicationby entering email & password	High	Jila Rasnat B Eisha V S
Sprint-1	Chatbot	USN-3	As a user I can ask query in chatbot.	Medium	Blessy Jefrina H Karthikeyan B
Sprint-2	Confirmation	USN-4	As a user, I can receive confirmation mail.	Medium	Lakshmi Meena M
Sprint-2	Dashboard	USN-5	As a user, I can view dashboard and select	Medium	Eisha V S
Sprint-2	View Donor List	USN-6	As a user, I can view all the donor list andcontact them directly	High	Jila Rasnat B
Sprint-2	Search Donor	USN-7	As a user, I can search for the donor	Medium	Karthikeyan B
Sprint-3	About us	USN-8	As a User, I can view the about us page which contains all contact information	Medium	Blessy Jefrina H
Sprint-3	Modify data	USN-9	As a admin, I can modify the User data.	High	Karthikeyan B
Sprint-3	Send mail	USN-10	As a user, I can send mail to donors using sendgrid.	High	Jila Rasnat B Lakshmi Meena M Eisha V S
Sprint-3	Home page	USN-11	As a user I can view the home page andselect the desired option.	Medium	Jila Rasnat B Lakshmi Meena M Eisha V S Blessy Jefrina H Karthikeyan B
Sprint-4	Send Query	USN-12	As a user I can ask my query through email.	Medium	Lakshmi Meena M Blessy Jefrina H
Sprint-4	Download data	USN-13	As a admin I can download the user data	High	Jila Rasnat B

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date(Planned)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	5 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	19 Nov 2022

6.3 Reports from JIRA





7. CODING AND SOLUTIONING

7.1 Feature 1

Python

- It is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.
- Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object- oriented and functional programming.

- Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support.
- Python consistently ranks as one of the most popular programming languages.

7.2 Feature 2

Flask

- Flask is a micro web framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries.
- It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.
- However, Flask supports extensions that can add application features as if they were implemented in Flask itself.
- Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

7.3 Feature 3

SendGrid

- SendGrid is a cloud-based SMTP provider that allows you to send email without having to maintain email servers.
- SendGrid provides two ways to send email: through our SMTP relay or through our Web API. SendGrid provides client libraries in many languages.
- This is the preferred way to integrate with SendGrid. If you choose to use SendGrid without a client library, the Web API is recommended in most cases as it is faster, provides some benefit with encoding, and tends to be easier to use.
- SMTP provides many features by default, but is harder to setup.

7.4 Database Schema

IBM Db2

- A hybrid ANSI-compliant data virtualization tool for accessing,querying and summarizing data across the enterprise which:

- Provides a massively parallel processing (MPP) architecture Exploits Hive, HBase and Apache Spark concurrently for best-in-class analytic capabilities
- Requires only a single database connection or query to connect disparate sources such as HDFS, RDMS, NoSQL databases, object stores andWeb HDFS
- Provides low latency support for ad-hoc and complex queries,high performance, and federation capabilities
- Understands dialects from other vendors and various productsfrom Oracle, IBM® Db2® and IBM Netezza®
- Enables advanced row and column security

Kubernetes

- Kubernetes — also known as “k8s” or “kube” — is a container orchestration platform for scheduling and automating the deployment,management, and scaling of containerized applications.
- Kubernetes was first developed by engineers at Google before being open sourced in 2014. It is a descendant of Borg, a container orchestrationplatform used internally at Google. Kubernetes is Greek
- for *helmsman* or *pilot*, hence the helm in the Kubernetes logo (link resides outside IBM).
- Today, Kubernetes and the broader container ecosystem are maturinginto a general-purpose computing platform and ecosystem that rivals — if not surpasses — virtual machines (VMs) as the basic building blocks of modern cloud infrastructure and applications.
- This ecosystem enables organizations to deliver a high- productivity Platform-as-a-Service (PaaS) that addresses multipleinfrastructure-related and operations-related tasks and issues
- surrounding cloud-native development so that development teams can focus solely on coding and innovation.

8. TESTING

8.1 Test Cases

- The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product.
- It provides a way to check the functional it your components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectation and does not fail in an unacceptable manner.
- There are various types of tests. Each test type addresses a specifictesting requirement

8.2 User Acceptance Testing

Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the Plasma Donor Application project at the time of the release to User Acceptance Testing (UAT).

Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	1	0	2
External	2	2	1	1	6
Fixed	4	1	1	10	16
Not Reproduced	0	0	0	0	0
Skipped	1	1	0	1	3

Won't Fix	0	2	2	0	4
Totals	24	14	13	26	51

Testcase Analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	9	0	0	9
Client Application	10	0	0	10
Security	1	0	0	1
Outsource Shipping	0	0	0	0
Exception Reporting	9	0	0	9
Final Report Output	9	0	0	9
Version Control	1	0	0	1

9. RESULTS

9.1 Performance Metrics

- Project metrics are used to track the progress and performance of a project.
- Monitoring parts of a project like productivity, scheduling, and scope make it easier for team leaders to see what's on track.
- As a project evolves, managers need access to changing deadlines or budgets to meet their client's expectations.

10. ADVANTAGES AND DISADVANTAGES

a) ADVANTAGES

- To supplement their body's clotting ability and stop excessive bleeding from occurring.
- The **COVID-19** pandemic severely impacted the availability of medicines made from plasma. To ensure there is never a shortage, we need your help.
- Help people by donating your plasma.
- Donating plasma stimulates blood cell production.
- May reduce the risk of heart attacks and other serious health issues.
- Simple user interface – very easy to use and understand. It is easily workable and accessible for everyone.
- It contains both the database of the donor as well as the recipients.
- It alleviates the burden of coordinator to manage users and resources easily.
- The application helps in finding the nearest donor in a very short time during emergency cases.
- It provides reassurance to patients concerned about blood risks.
- A plasma can contain antibodies to the infection. If another person receives this plasma, it may help their body fight the virus.
- Plasma donation offers many health benefits with few risks.

b) DISADVANTAGES

- Possible side effects of donating plasma include dehydration, vein damage, fainting, and fatigue.

- After plasma donation, some person can experience reduction in plasma levels and increase in infection risk.
- The process involves puncturing of the veins with a 17 or 16-gage needle, which are sufficient to prevent vein damage, except with improper handling.
- Unregulated donation is risky to both donors and recipients.
- Unhealthy donors become more prone to disease and infection, and may even at risk of death.
- It depletes the calcium levels in the body.
- People who have recovered from COVID-19 may be able to help others with the disease by donating blood plasma, according to the Food and Drug Administration (FDA).
- This application requires an internet connection for the working of the website.
- Auto-Verification - It cannot automatically verify the genuine user.
- Does not affect risk of ABO incompatibility.
- It increases prevalence of adverse reactions to autologous donation.

11. CONCLUSION

Blood's liquid component, called plasma, is made up of a combination of salts, proteins, and water. Proteins generated by are called antibodies, the reaction of the body to an infection. Plasma donations are encouraged for COVID19 survivors who have been totally rescued which can lengthen the lives of additional individuals since their plasma includes antigens which helps in speedier recovery for the affected person. This model is made user friendly so anybody can view and maintain his/her account. This project will help new blood/plasma banks improve their services and progress from traditional to user-friendly frameworks. Symptoms for at least 14 days before to the donation are required. The efficient way of finding plasma donor for the infected people is implemented using the plasma donor website that is hosted using SendGrid and IBM database. To ensure the smooth functioning of the website operations. Our sole purpose is to reduce the time as well as the sufferings of the people. A blood or plasma recipient can efficiently get donors using our system in any situation. This model is made user friendly so anybody can view and maintain his/her account. This application will break the chain of business through blood/plasma and help the poor to find donor at free of cost. This project will

help new blood/plasma banks improve their services and progress from traditional to user-friendly frameworks.

12. FUTURE SCOPE

- Upgrading the UI that is more user-friendly which will help many users to access the website and also ensures that many plasma donors can be added into the community.
- Using elastic load balancer, it helps to handle multiple requests at the same time which will maintain the uptime of the website with negligible downtime.
- Ensures the faster and efficient communication between the donor and the recipients.
- Due to the sensitivity of the profession, the salary offers for healthcare data analysts are lucrative around the world.
- The future of big data in healthcare will be determined by technological breakthroughs from 2022 to 2030.

13. APPENDIX

13.1 Source Code

```
import ibm_db

from flask import *
from flask_mail import Mail, Message
app = Flask(__name__)
mail = Mail(app) # instantiate the mail class

# configuration of mail
app.config['MAIL_SERVER']='smtp.gmail.com'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USERNAME'] = 'karthikeyan.b2019@kgkite.ac.in'
app.config['MAIL_PASSWORD'] = '*************'
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True
mail = Mail(app)
donor_vs_patient_compatability = {
    'O+':"('O+','O-')",
    'O-':"('O+','O-')",
```

```
    "A+":"('O+','A+','O-','A-')",
    "A-":"('O+','A+','O-','A-')",
    "B+":"('O+','B+','O-','B-')",
    "B-":"('O+','B+','O-','B-')",
    "AB+":"('O+','A+','B+','AB+','O-','A-','B-','AB-')",
    "AB+":"('O+','A+','B+','AB+','O-','A-','B-','AB-')",
}
patient_vs_donor_compatability = {
    "O+":"('O+','A+','B+','AB+','O-','A-','B-','AB-')",
    "O-":"('O+','A+','B+','AB+','O-','A-','B-','AB-')",
    "A+":"('A+','AB+','A-','AB-')",
    "A-":"('A+','AB+','A-','AB-')",
    "B+":"('B+','AB+','B-','AB-')",
    "B-":"('B+','AB+','B-','AB-')",
    "AB+":"('AB+','AB-')",
    "AB-":"('AB+','AB-')",
}
@app.route('/')
def home():
      return render_template('index.html')
@app.route('/sign_up')
def signUp():
      return render_template('sign_up.html')
@app.route('/sign_in')
def signIn():
      return render_template('sign_in.html')
@app.route('/request')
def requests():
      email = request.cookies.get('email')
      name = request.cookies.get('name')
      if email != None:
            resp = make_response(render_template('request.html',email = email,
name = name, logged_in = True))
      else:
            resp = make_response(render_template('request.html',email = email,
name = name, logged_in = False))
      return resp
@app.route('/donor_registration')
def donor_registration():
      email = request.cookies.get('email')
      name = request.cookies.get('name')
      isDonor = False
      if email != None:
            conn = ibm_db.connect(
```

```
                        'DATABASE=bludb;HOSTNAME=b1bc1829-6f45-4cd4-bef4-
10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32304;SECURITY=S
SL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=gfn00031;PWD=LITZUQj2tpFc3t0
i', '', '')
            sql = 'select * from donors where email='+'\''+email+'\''
            stmt = ibm_db.exec_immediate(conn, sql)
            dictionary = ibm_db.fetch_assoc(stmt)
            isDonor = False
            if dictionary != False:
                  isDonor = True
      if isDonor:
            resp = make_response(render_template('donor_registration.html',email
= email, name = name, isDonor = True, logged_in = True))
      elif email != None:
            resp = make_response(render_template('donor_registration.html',email
= email, name = name, logged_in = True))
      else:
            resp = make_response(render_template('donor_registration.html',email
= email, name = name, logged_in = False))
      return resp
@app.route('/add_user', methods=['POST', 'GET'])

def add_user():

    if request.method == 'POST':
        try:
            name = request.form['name']
            email = request.form['email']
            password = request.form['pass']
            conn = ibm_db.connect(
                'DATABASE=bludb;HOSTNAME=b1bc1829-6f45-4cd4-bef4-
10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32304;SECURITY=S
SL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=gfn00031;PWD=LITZUQj2tpFc3t0
i', '', '')
            sql = "select * from users where email = "+"'"+email+"'"
            stmt = ibm_db.exec_immediate(conn, sql)
            user = ibm_db.fetch_assoc(stmt)
            if user:
                  msg = "Account already exists"
            else:
                  sql = "insert into users values(?,?,?)"
                  param = name, email, password,
                  stmt = ibm_db.prepare(conn, sql)
                  ibm_db.execute(stmt, param)
                  msg = "You're successfully signed up!"
```

```
                    mail_msg = Message(
                    'Welcome to Planor',
                    sender ='karthikeyan.b2019@kgkite.ac.in',
                    recipients = [email]
                    )
                    mail_msg.body = "Hi "+name+" you have successfully signed up
into planor."
                    mail.send(mail_msg)
        except Exception as e:
            print("exception occured!",e)
            msg = e
        finally:
            return render_template('post_signup.html', msg = msg)
@app.route('/validate_user',methods = ['POST', 'GET'])
def validate_user():
   if request.method == 'GET':
      try:
            args = request.args
            email = args.get('email')
            password = args.get('password')
            conn = ibm_db.connect(
                'DATABASE=bludb;HOSTNAME=b1bc1829-6f45-4cd4-bef4-
10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32304;SECURITY=S
SL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=gfn00031;PWD=LITZUQj2tpFc3t0
i', '', '')
            sql = 'select * from users where email='+'\''+email+'\''
            stmt = ibm_db.exec_immediate(conn, sql)
            dictionary = ibm_db.fetch_assoc(stmt)
            print("executed")
            print(dictionary)
            if dictionary != False:
                        if(dictionary["PASSWORD"]== password):
                               print("success")
                               resp =
make_response(render_template("post_signin.html"))
                               resp.set_cookie('email', dictionary["EMAIL"])
                               resp.set_cookie('name',dictionary["NAME"])
                               print("success")
                               return resp
                        else:
                              return "Incorrect Password"
            else:
                  return "User does not exists"

      except Exception as e :
```

```
        print("error",e)
        return repr(e)
@app.route('/add_donor', methods=['POST', 'GET'])
def add_donor():
    if request.method == 'POST':
        try:
            name = request.form['name']
            email = request.form['email']
            blood_group = request.form['blood_group']
            contact_no = request.form['contact_no']
            location = request.form['city']
            conn = ibm_db.connect(
                'DATABASE=bludb;HOSTNAME=b1bc1829-6f45-4cd4-bef4-
10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32304;SECURITY=S
SL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=gfn00031;PWD=LITZUQj2tpFc3t0
i', '', '')
            sql = "insert into donors values(?,?,?,?,?)"
            param = name, email,blood_group,contact_no, location,
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.execute(stmt, param)
            msg = "You're successfully registered as donor"
        except Exception as e:
            print("exception occured!",e)
            msg = e
        finally:
            return render_template('donor_registration_status.html', msg = msg)
@app.route('/create_request', methods=['POST', 'GET'])
def create_request():
    if request.method == 'POST':
        try:
            name = request.form['name']
            email = request.form['email']
            blood_group = request.form['blood_group']
            contact_no = request.form['contact_no']
            location = request.form['city']
            request_status =  "Pending"
            conn = ibm_db.connect(
                'DATABASE=bludb;HOSTNAME=b1bc1829-6f45-4cd4-bef4-
10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32304;SECURITY=S
SL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=gfn00031;PWD=LITZUQj2tpFc3t0
i', '', '')
            sql = "insert into requests (name, email, blood_group, contact_no,
location, request_status) values(?,?,?,?,?,?)"
            param = name, email,blood_group,contact_no, location, request_status,
            stmt = ibm_db.prepare(conn, sql)
```

```
            ibm_db.execute(stmt, param)
            msg = "You're successfully made a request!"
            sql = "select email from donors where blood_group in
"+patient_vs_donor_compatability[blood_group]+"and location= "+"'"+location+"'"
            print(sql)
            stmt = ibm_db.exec_immediate(conn, sql)
            donor_mails = []
            dictionary = ibm_db.fetch_assoc(stmt)
            while dictionary != False:
                  donor_mails.append(dictionary["EMAIL"])
                  dictionary = ibm_db.fetch_assoc(stmt)
            mail_msg = Message(
                'New Plasma Request Received',
                sender ='karthikeyan.b2019@kgkite.ac.in',
                recipients = donor_mails
               )
            mail_msg.body = "Hello, A new request has been received. Kindly check
it out!\nRequester mail id: "+email
            mail.send(mail_msg)
        except Exception as e:
            print("exception occured!",e)
            msg = e
        finally:
            return render_template('donor_registration_status.html', msg =
msg)
@app.route('/accept_request', methods=['POST', 'GET'])
def accept_request():
    if request.method == 'POST':
        try:
            id = request.form['id']
            email = request.cookies.get('email')
            conn = ibm_db.connect(
                'DATABASE=bludb;HOSTNAME=b1bc1829-6f45-4cd4-bef4-
10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32304;SECURITY=S
SL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=gfn00031;PWD=LITZUQj2tpFc3t0
i', '', '')
            sql = "update requests set request_status = 'Accepted' , accepted_by
="+"'"+email+"'"+"where id ="+"'"+id+"'"
            stmt = ibm_db.exec_immediate(conn, sql)

            #retrieving recipient contact details
            sql = 'select * from requests where id='+'\''+id+'\''
            stmt = ibm_db.exec_immediate(conn, sql)
            recipient_details = ibm_db.fetch_assoc(stmt)
            #retrieving donor contact details
```

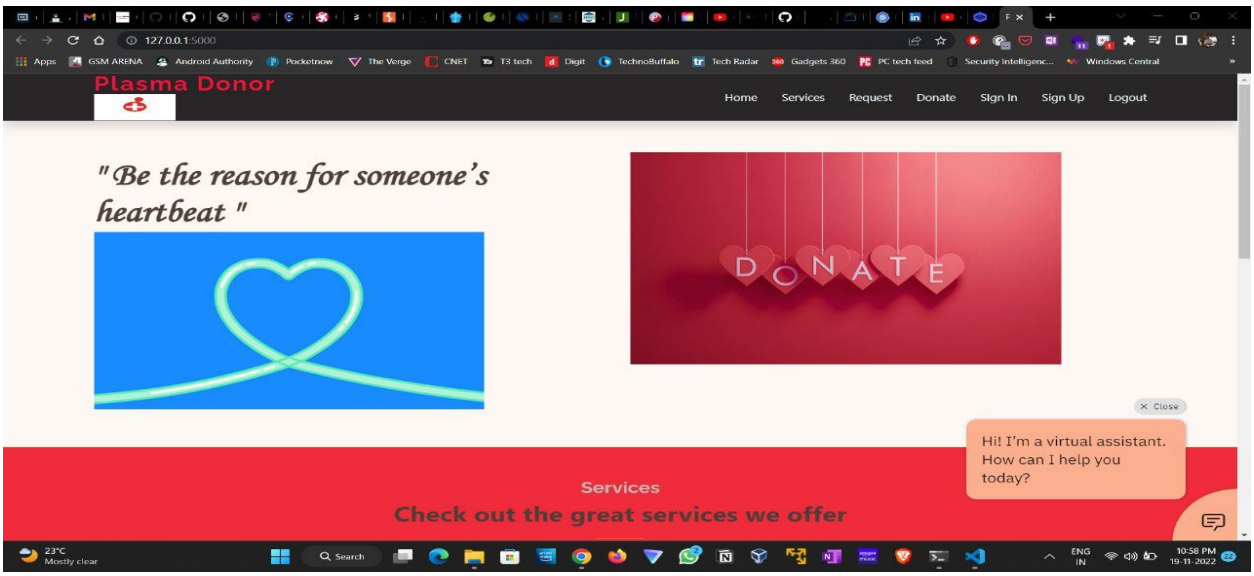
```
            sql = 'select * from donors where email='+'\''+email+'\''
            stmt = ibm_db.exec_immediate(conn, sql)
            donor_details = ibm_db.fetch_assoc(stmt)
            mail_msg = Message(
                'Request Accpeted',
                sender ='karthikeyan.b',
                recipients = [recipient_details["EMAIL"]]
               )
            mail_msg.body = "Hi "+recipient_details["NAME"]+" you request has
been accepted by "+donor_details["NAME"]+" \nContact Details:\nEmail:
"+donor_details["EMAIL"]+"\nContact No.: "+donor_details["CONTACT_NO"]
            mail.send(mail_msg)
            mail_msg = Message(
                'Recipient Details',
                sender ='19eucs180@skcet.ac.in',
                recipients = [donor_details["EMAIL"]]
               )
            mail_msg.body = "Hi "+donor_details["NAME"]+". Request ID: "+id+" has
been accepted by you\nContact Details of the recipient:\nName:
"+recipient_details["NAME"]+"\nEmail: "+recipient_details["EMAIL"]+"\nContact
No.: "+recipient_details["CONTACT_NO"]
            mail.send(mail_msg)
        except Exception as e:
            print("exception occured!",e)
        finally:
            return redirect(url_for('profile'))
@app.route('/profile')
def profile():
      email = request.cookies.get('email')
      name = request.cookies.get('name')
      if email != None:
            conn = ibm_db.connect(
                  'DATABASE=bludb;HOSTNAME=b1bc1829-6f45-4cd4-bef4-
10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32304;SECURITY=S
SL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=gfn00031;PWD=LITZUQj2tpFc3t0
i', '', '')
            sql = 'select * from requests where email='+'\''+email+'\''
            stmt = ibm_db.exec_immediate(conn, sql)
            requests = []
            dictionary = ibm_db.fetch_assoc(stmt)

            while dictionary != False:
                  print(dictionary["ID"])
                  requests.append(dictionary)
                  dictionary = ibm_db.fetch_assoc(stmt)
```

```
            print(requests)
            sql = 'select * from donors where email='+'\''+email+'\''
            stmt = ibm_db.exec_immediate(conn, sql)
            dictionary = ibm_db.fetch_assoc(stmt)
            isDonor = False
            pending_requests = []
            if dictionary != False:
                  isDonor = True
                  donor_location = dictionary["LOCATION"]
                  donor_bloodgroup = dictionary["BLOOD_GROUP"]
                  sql = "select * from requests where
blood_group="+"'"+donor_bloodgroup+"'"+"and location=
"+"'"+donor_location+"'"+"and request_status= "+"'Pending'"
                  stmt = ibm_db.exec_immediate(conn, sql)
                  dictionary = ibm_db.fetch_assoc(stmt)
                  while dictionary != False:
                        pending_requests.append(dictionary)
                        dictionary = ibm_db.fetch_assoc(stmt)
                  print(pending_requests)
            accepted_requests= []
            if isDonor:
                  sql = 'select * from requests where
accepted_by='+'\''+email+'\''
                  stmt = ibm_db.exec_immediate(conn, sql)
                  dictionary = ibm_db.fetch_assoc(stmt)
                  while dictionary != False:
                        accepted_requests.append(dictionary)
                        dictionary = ibm_db.fetch_assoc(stmt)
                  print(accepted_requests)
            return render_template('profile.html', name =name, email =
email,requests_len = len(requests) ,requests = requests, pending_requestslen =
len(pending_requests), pending_requests = pending_requests, accepted_requestslen
= len(accepted_requests), accepted_requests = accepted_requests, logged_in=True)
      else:
            return render_template('profile.html', logged_in= False)
@app.route('/stats')
def stats():
            conn = ibm_db.connect(
                        'DATABASE=bludb;HOSTNAME=b1bc1829-6f45-4cd4-bef4-
10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32304;SECURITY=S
SL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=gfn00031;PWD=LITZUQj2tpFc3t0
i', '', '')
            sql = 'select count(email) from donors'
            stmt = ibm_db.exec_immediate(conn, sql)
            donors= ibm_db.fetch_assoc(stmt)["1"]
```

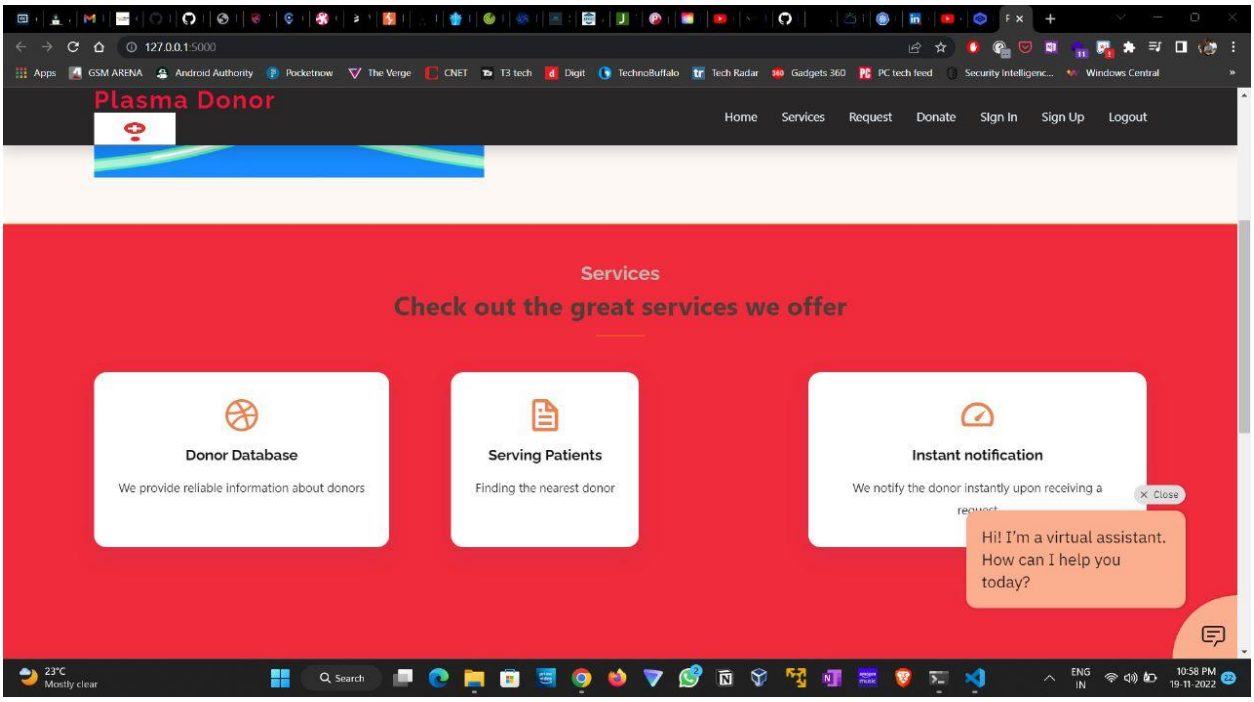
```
            sql = "select count(email) from requests where request_status =
'Pending'"
            stmt = ibm_db.exec_immediate(conn, sql)
            pending_requests= ibm_db.fetch_assoc(stmt)["1"]
            sql = "select count(email) from requests where request_status =
'Accepted'"
            stmt = ibm_db.exec_immediate(conn, sql)
            accepted_requests= ibm_db.fetch_assoc(stmt)["1"]
            return render_template('stats.html', donors = donors,
pending_requests = pending_requests, accepted_requests = accepted_requests)
@app.route('/logout')
def logout():
      email = request.cookies.get('email')
      if email != None:
            resp = make_response(render_template('logout.html',loggedin = True))
            resp.set_cookie('name', '', expires=0)
            resp.set_cookie('email', '', expires=0)
      else:
            resp = make_response(render_template('logout.html',loggedin = False))
      return resp
if __name__ == '__main__':
      app.run(debug=True)
```

Output:

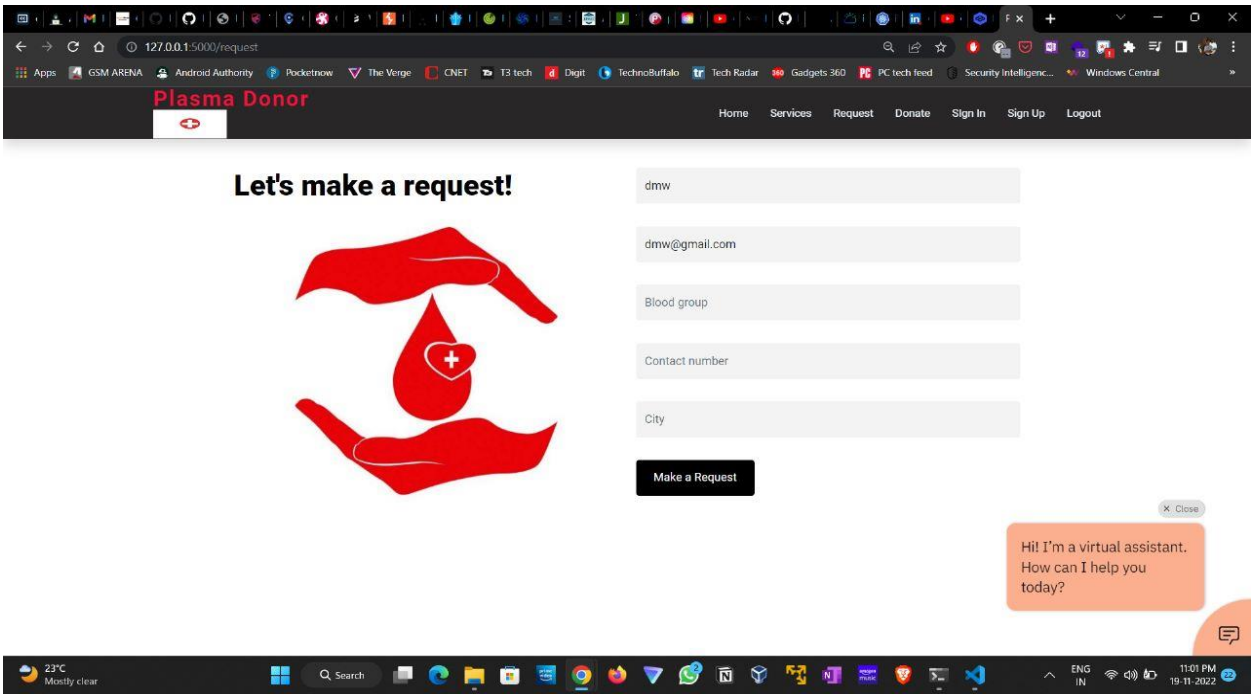
Home



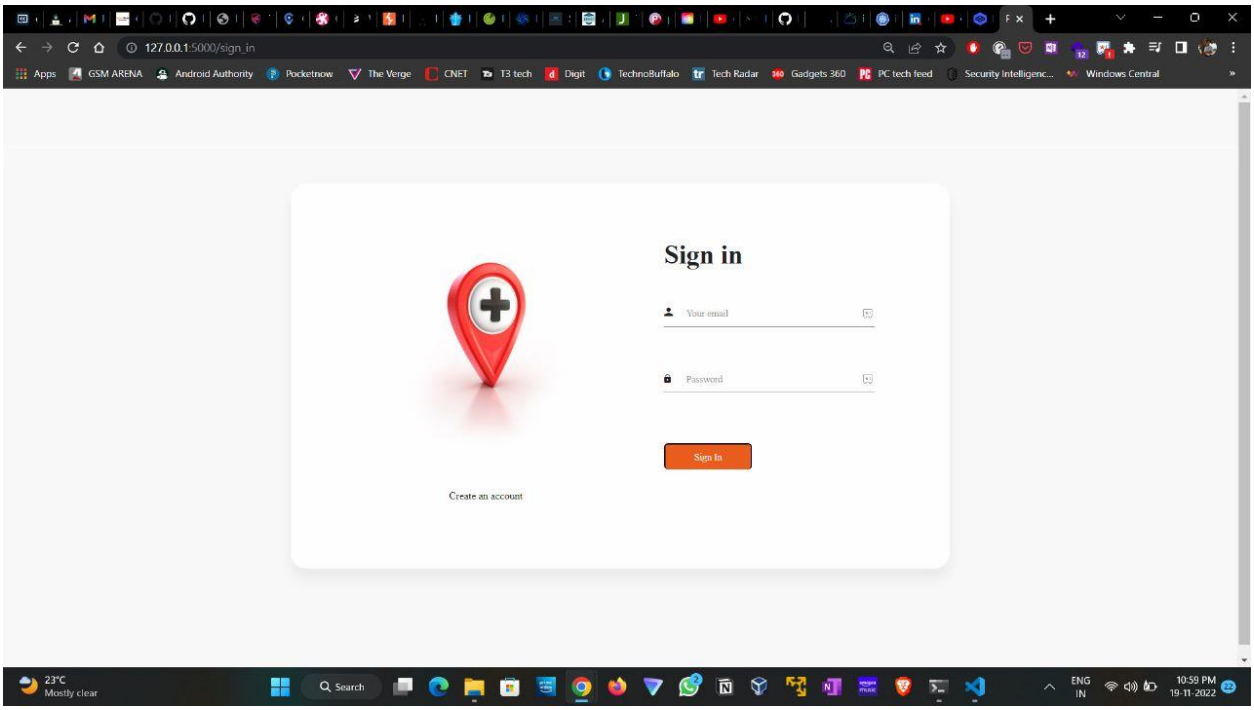
Services



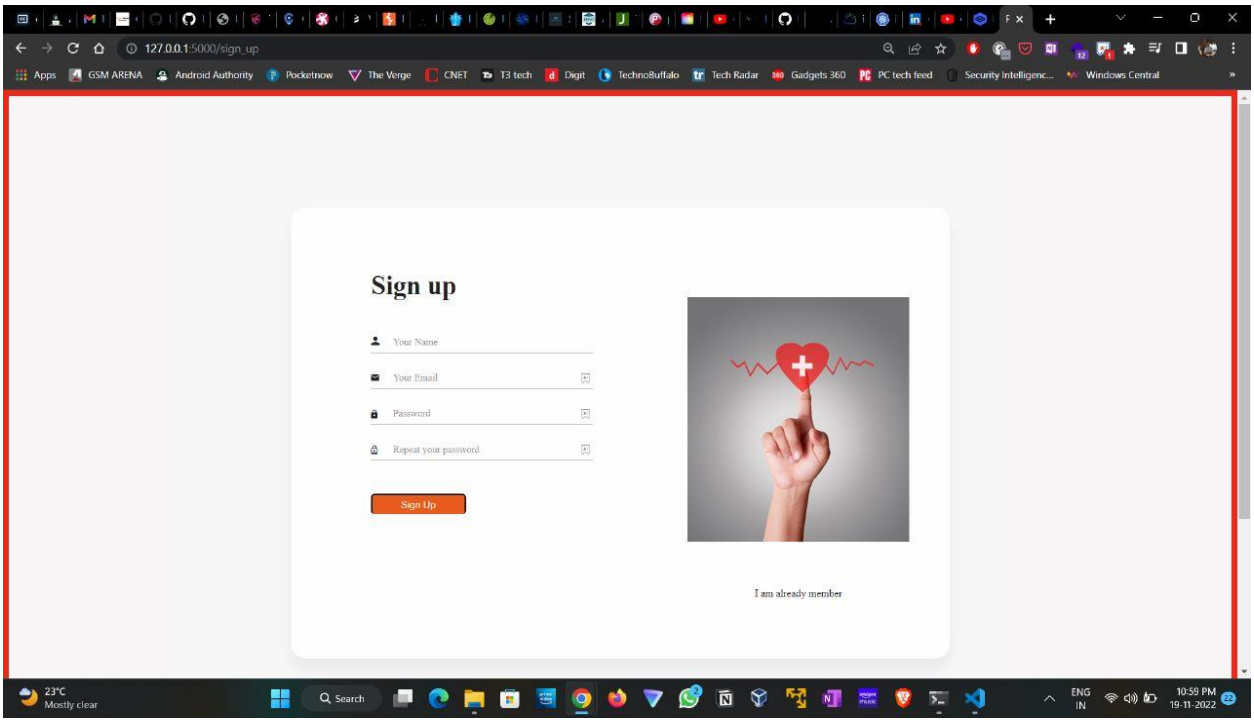
Request



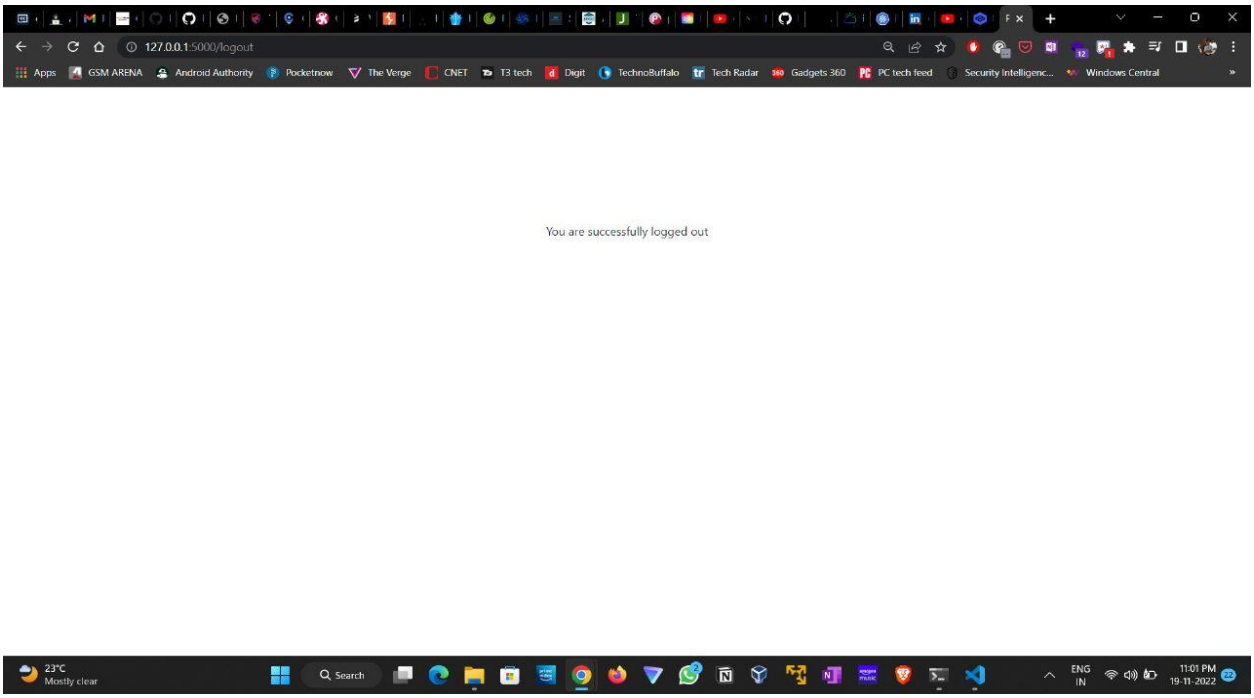
Sign in



Sign up



Log out



13.2 GitHub And Project Demo Link

a) GitHub

https://github.com/IBM-EPBL/IBM-Project-46664-1660752877

b) Project Demo Link

https://drive.google.com/file/d/1lQtDscYbafX64w1qy9YkOSKXJD6IUE3D/view